AMENDMENTS TO THE CLAIMS

Please amend the claims as indicated in the following listing of all claims:

1. (Original) A method of generating object lifetime statistics based on run-time observations, the method comprising:

selecting from amongst object instances of an observed category, a sampled subset of the object instances allocated in one or more execution threads of a computational system;

coincident with allocation of a sampled instance of an object, establishing a weak reference thereto and associating therewith information indicative of at least allocation time; and

referencing the sampled instances at run-time via the weak references and updating the object lifetime statistics based on the associated allocation time and then-current state.

2. (Original) The method of claim 1,

wherein the then-current state includes garbage collection state of sampled instances.

3. (Original) The method of claim 1,

wherein the computational system includes a garbage collector; and

wherein the object lifetime statistics updating is performed in response to a determination by the garbage collector that one or more sampled instances have become unreachable.

4. (Original) The method of claim 1,

wherein the computational system includes a generational garbage collector; and

wherein the object lifetime statistics updating is performed in response to a determination by the generational garbage collector that one or more sampled instances have become unreachable or have been promoted from a younger generation to an older generation.

5. (Original) The method of claim 1,
wherein the object lifetime statistics updating is performed by periodically accessing the sampled instances.

6. (Original) The method of claim 1,
wherein the object lifetime statistics updating is performed by purging a subset of the object lifetime statistics.

7. (Original) The method of claim 1,
wherein the object lifetime statistics are represented as histogram of lifetimes.

8. (Previously presented) The method of claim 1,
wherein the object lifetime statistics are represented as mean lifetimes.

9. (Previously presented) The method of claim 1,
wherein the object lifetime statistics are calculated using an average birth date.

10. (Original) The method of claim 1,
wherein an observed category corresponds to an object class.

11. (Original) The method of claim 1,
wherein an observed category corresponds to a garbage collection generation.

12. (Original) The method of claim 1,
wherein the associated information indicative of at least allocation time is further indicative of allocation site.

13. (Original) The method of claim 1,
wherein the associated information indicative of at least allocation time is further indicative of an allocating one of the execution threads.

14. (Original) The method of claim 1,

wherein the associated information indicative of allocation time is encoded as one or more of allocation count, system time, CPU time, byte count, and garbage collection count.

15. (Original) The method of claim 1,
wherein the weak reference is of a type not considered in reachability analysis of a garbage collector.

16. (Original) In an automatically reclaimed storage environment, a method of sampling instances of software objects during respective lifetimes thereof, the method comprising:
establishing weak references to respective of the sampled instances, each of the weak references identifying at least one respective sampled instance;
associating allocation-time information with each sampled instance; and
accessing the sampled instances via the weak references and performing an action based at least in part on a state of one or more of the sampled instances and respective allocation-time information.

17. (Original) The method of claim 16,
wherein the weak reference establishing includes storing in a data structure a reference not considered in reachability analysis of the automatically reclaimed storage environment.

18. (Original) The method of claim 16,
wherein the sampled instances include a representative subset of a category of software objects.

19. (Original) The method of claim 18,
wherein the category is object class specific.

20. (Original) The method of claim 18,
wherein the category is call-site specific.

21. (Original) The method of claim 18,
wherein the category corresponds to an activation record stack profile.

22. (Original) The method of claim 18,
wherein the category covers an abstract class or interface.

23. (Original) The method of claim 18,
wherein the category is specific to a particular garbage collection space.

24. (Original) The method of claim 16, wherein the allocation-time information includes one or more of:
time of allocation;
allocation site;
allocating thread; and
object type.

25. (Original) The method of claim 16, further comprising:
selecting at allocation time the sampled instances from amongst all instances of a particular type.

26. (Original) The method of claim 25,
wherein the selecting is based on allocation buffer overflow.

27. (Original) The method of claim 25,
wherein the selecting is based on a subset of allocations for each type of sampled software object.

28. (Original) The method of claim 27,
wherein the subset includes a pseudo random distribution of the allocations.

29. (Original) The method of claim 27,
wherein the subset includes a deterministic distribution of the allocations.

30. (Currently Amended) An object sampling facility for a computational system, the object sampling facility comprising:

a weak reference construct implemented by the computational system; and

an object fingerprinter responsive to a storage allocator of the computational system, the object ~~finger printer~~ fingerprinter associating (1) allocation-time information and (2) an instance of the weak reference construct with at least a sampled subset of objects allocated by the storage allocator.

31. (Original) The object sampling facility of claim 30, further comprising:

an object sampler responsive to garbage collection events in the computational system, the object sampler referencing the sampled subset via the weak reference instances and maintaining object lifetime statistics based on the associated allocation-time information and then-current state of the sampled subset.

32. (Original) The object sampling facility of claim 30, further comprising:

an object sampler referencing the sampled subset via the weak reference instances and maintaining object lifetime statistics based on the associated allocation-time information and sampled state of the sampled subset.

33. (Original) The object sampling facility of claim 32,

wherein the storage allocator is responsive to the object lifetime statistics in its allocation decisions.

34. (Cancelled)

35. (Currently Amended) The object sampling facility of claim ~~3~~432,

wherein a generational garbage collector is responsive to the object lifetime statistics in its promotion decisions.

36. (Original) The object sampling facility of claim 30, wherein the allocation-time information is indicative of one or more of:

a time of object allocation;

an allocation site; and

an allocating thread.

37. (Original) The object sampling facility of claim 30, embodied as a computer program product.

38. (Currently Amended) A computer program product encoded in at least one computer readable medium, the computer program product comprising:

at least one functional sequence for associating allocation-time information and an instance of a weak reference with at least a sampled subset of objects allocated by a storage allocator; and

at least one functional sequence for sampling the sampled subset using the weak reference instances and maintaining object lifetime statistics based on the associated allocation-time information and sampled state of the sampled subset.

39. (Original) A computer program product as recited in 38, embodied as a generational garbage collector and further comprising:

at least one functional sequence for tenuring certain object instances in accordance with those of the object lifetime statistics corresponding thereto.

40. (Original) A computer program product as recited in 38, embodied as a generational run-time profiler.

41. (Original) A computer program product as recited in 38,

wherein the at least one computer readable medium is selected from the set of a disk, tape or other magnetic, optical, or electronic storage medium and a network, wireline, wireless or other communications medium.

42. (Original) An apparatus comprising:

means for associating allocation-time information with sampled instances of software objects;

means for referencing the sampled instances of software objects, the referencing means operable for both reachable and unreachable ones thereof;

means for updating lifetime predictions for categories of the software objects based on run-time access to states of corresponding ones of the sampled instances and associated allocation-time information therefor.

43. (Previously presented) The method of claim 1,
wherein at least some of the sampled subset of the object instances are sampled coincident with death of respective ones of the sampled memory objects.

44. (Previously presented) The method of claim 16,
wherein at least some of the sampling is performed coincident with death of respective ones of the sampled instances.

45. (Previously presented) The object sampling facility of claim 30,
wherein at least some of the sampled subset of objects are sampled coincident with death of respective ones of the sampled objects.

46. (Previously presented) A computer program product as recited in claim 38,
wherein at least some of the sampling is performed coincident with death of respective ones of the sampled objects.

47. (Previously presented) The apparatus of claim 42,
wherein at least some of the sampled instances are sampled coincident with death of respective ones of the sampled instances.